(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 109 567**
B1

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication of patent specification: **08.10.86**

(51) Int. Cl.⁴: **G 06 F 9/44**

(21) Application number: **83110498.9**

(22) Date of filing: **21.10.83**

(54) Accelerated instruction mapping external to source and target instruction streams for near realtime injection into the latter.

(30) Priority: **22.10.82 PCT/US 82/01496**

(43) Date of publication of application:
30.05.84 Bulletin 84/22

(45) Publication of the grant of the patent:
08.10.86 Bulletin 86/41

(84) Designated Contracting States:
DE FR GB IT

(56) References cited:
GB-A-2 002 553
US-A-4 315 321

IBM TECHNICAL DISCLOSURE BULLETIN, vol. 15, no. 3, August 1972, pages 920-921, New York, USA, J.C. KEMP: "Instruction translator"

(73) Proprietor: **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504 (US)

(72) Inventor: **Fisk, Dale Edward**
1704 Husted Avenue
San Jose California 95124 (US)
Inventor: **Griffith, Robert Leroy**
5995 Alvarado Court
San Jose California 95120 (US)
Inventor: **Homan, Merle Edward**
236 Old Adobe Road
Los Gatos California 95030 (US)
Inventor: **Radin, George**
26 Franklin Street
Piermont New York 10968 (US)
Inventor: **Richards, Waldo**
7167 Via Romero
San Jose California 95139 (US)

(74) Representative: **Grant, Iain Murray**
IBM United Kingdom Limited
Intellectual Property Department
Hursley Park
Winchester Hampshire SO21 2JN (GB)

Courier Press, Leamington Spa, England.

EP 0 109 567 B1

Description

This invention relates to a new facility realtime format conversion of multifield instructions for a CPU of a first kind to multifield instructions for a CPU of a second kind by facilities external to the CPU of the second kind and without disrupting the logical flow or execution of either source or target instruction streams.

Background Art

Parks et al, U.S. Patent 4,315,321, "Method and Apparatus for Enhanching the Capabilities of a Computing System", issued 9 February 1982, teaches that indicator codes within an activation record can select one of several mutually exclusive microcode sets for interpreting a referenced instruction stream. The activation record is manifest in the form of program status word register bit positions. Parks is concerned with interpreting or construing the intent of each referenced main memory instruction. Further, the activation record of the process external to the instruction provides the microcode set selection index.

Nutter, U.S. Patent 3,543,245 "Computer Systems", issued 24 November 1970, asserts that if control words, indexed by the OP code portion of a CPU multifield instruction, are used to select the instruction fields and their microcode execution order, then varying instruction widths can be accommodated. Nutter, as does Parks, is concerned with the CPU executing the intent of an external instruction by way of immediate microcode interpretation. Nutter's control words include masks and switching and shifting cirucit control bits by which fields in a source instruction would be selected and reordered to form a "queue" for immediate microcode execution. Indeed, the field selection is described from the specification, column 6, line 23, through column 10, line 54, while the mapping of randomly ordered fields in the source word into predetermined positions in a target word is described at column 62, lines 5 through 36. Further, Nutter shows a control word register pair of FIG. 2 as a single register equivalent in FIG. 38. These are discussed at column 24, line 60 through column 31, line 30 in FIGS. 2, 9, and 37.

Other pertinent references include Cassonnet et al, U.S. Patent 3,997,895, "Date Processing System with a Microprogrammed Dispatcher for Working Either in Native or Non-native Mode", issued 14 December 1976, and Malcolm et al, U.S. Patent 3,698,007, "Central Processor Unit Having Simulative Interpretation Capability", issued 10 October 1972. Cassonet depicts a microprogrammable switch (130) responsive to preselected bit position content in an external instruction for having control stored microcode sequences interpreted respectively by the arithmetic logic unit (ALU 1317) or emulator unit (EMU 1316). Malcolm uses the OP code of the simulated instruction as an index into a set of simulator routines, and provides for storage of a base address to which the OP code index is an offset. Lastly, each instruction references only one operand. This configuration directly executes the intent of the non-native instructions.

A class of VLSI implementable computers with reduced instruction sets being driven by a respective data stream and instruction stream from corresponding caches has been described by Radin in "The 801 Minicomputer", appearing in the ACM Proceedings of the Symposium on Architectural Support for Programming Languages and Operating Systems", March 1—3, 1982, in Palo Alto, California, at pages 39—47. A similar CPU architecture was described by Patterson and Sequin in "RISC 1: a Reduced Instruction Set VLSI Computer", in the IEEE 8th Annual Symposium on Architecture Conference Proceedings of May 12—14, 1981, at pages 443—449, and in expanded form in IEEE Computer, September 1982 at pages 8—20. In this type of machine, instructions are obtained from an "Instruction Cache", and data is obtained from a separate (data cache), both of which are managed by a LRU information algorithm. Thus, all frequently used functions and data are likely to be found in their respective cache.

IBM Technical Disclosure Bulletin, Vol. 15, No. 3, August 1972, page 920, discloses a simple hardware instruction translation assist which is designed to sit in the instruction stream of a host or target processor and, by a table look-up process and selective register to register gating under control of the table look-up, to translate instructions in a fixed format into instructions in a second fixed format. Essentially, the OPCODE is replaced on a one-to-one basis and the remainder of the instruction is simply re-arranged. Such an arrangement, while self-controlling, fails if either the required translation is from one instruction to more than one instruction or if the final instruction stream that is required to be executed is mixed source and target instructions.

In contrast, the present invention provides apparatus for translating multifield source instructions into multifield target instructions in a target processor capable of executing target instructions and not source instructions by means of a self-controlling instruction translator logically situated between the instruction source and the target processor instruction execution element, characterised in that

a) the translator is micro-instruction controlled;

b) the translator is arranged to fetch a skeleton target instruction and one, or, sequentially, more than one, micro-instruction from memory from a location indexed by a pre-determined field of the source instruction currently being translated;

c) the translator is arranged, under control of the micro-instruction(s), to fill-in the skeleton target instruction by copying selected fields from the source instruction or by generating fields by computation from selected fields of the source instruction; and

d) the translator is arranged to insert the skeleton target instruction, once filled-in, into the target processor instruction stream.

Each fetched micro-instruction may include an address portion which, if non-zero, designates the memory location of the next micro-instruction of a sequence of related micro-instructions or, if zero, indicates termination of the sequence and the translator can be so arranged that the aforesaid operations of fetching, filling-in and inserting can be performed in an over-lapped manner.

Where the target processor includes two caches in its memory system, the first of the caches containing data, from the point of view of the target processor, and the second containing instructions, the translator may be arranged to fetch source instructions from the first cache and micro-instructions and skeleton instructions from the second cache, using cycle steal access to the second cache and to the instruction execution cycles of the execution element of the target processor.

It is observed that it is an object of this invention to convert multifield source instructions into multifield target machine instruction stream without otherwise perturbing the normal target machine instruction execution sequence. It is a related object to devise an efficient method of mapping the register space and constants of the source instruction set into that of the target wherein the method does not participate itself in the execution of these instructions. It is still a further object that such a conversion be executed external to the target machine and in near real-time, permitting the target machine to participate in emulations without itself being substantially modified.

The invention is predicated on a number of unexpected observations. These are (1) if a data stream comprising multiple field source machine instructions is mapped into the instruction stream of the target machine by an interposed independent processor, it enhances full realtime utilization due to the independent overlapping of such conversion; (2) if the preponderance of the source instruction fields can be used unchanged in the target machine instructions, then reformatting within an independent processor can be implemented by register-to-register transfers; (3) if a predetermined field within a source instruction indexes and accesses a word pair from memory, and if one word of the pair is a control section designating the field-to-field (register-to-register) mapping, and if the other word of the pair is a skeleton target instruction, it can be filled out by the fields of the source instruction; and (4) if target machine instructions are constructed external to said target machine, then the target machine is less complex and admits faster instruction execution.

Brief Description of the Drawings

FIG. 1 depicts the fields of an IBM 370 CPU instruction and its general mapping relation to a target machine instruction;

FIG. 2 depicts the emulator-assist processor (EAP) of the invention in communicating relation with the instruction and data caches of the target machine;

FIG. 3 sets out a bare relation of the source multifield IBM 370 CPU instruction and the micro-instruction, including the skeleton target instruction to be fleshed out through the mapping logic contained within the EAP;

FIG. 4 shows a definition of a microinstruction control section used by the EAP in fleshing out a skeleton target machine instruction;

FIG. 5 is a timing diagram of the major reformatting operations of the EAP; and

FIG. 6 is a completed field register definition of the EAP set out in FIG. 3.

Best Mode for Carrying out the Invention

While the invention does not reside in the architecture of either the target or source CPU instruction stream generators or receivers, the target CPU does serve as the environment within which the invention is practised. As the aforementioned Radin and Patterson references exemplify, the new trend in CPU architecture is the use of a reduced instruction set and of independently pipelined instruction and data streams terminating in said CPU. For many years instruction and data reference speeds have been increased by use of least recently used (LRU) managed information caches between the CPU main memory and the target CPU. Thus, the immediately referenced instruction stream is resident in one cahce while the immediate reference data stream is referenced in a second. Such a target CPU is shown in FIG. 2.

Typically, the target machine (CPU) 1 is organized to permit independent memory access for the data and instructions. Each access path is served by an independent cache. Thus, instruction cache 5 is accessed by address line 9 with the information therefrom being read over path 11, 13, and 21. Likewise, data cache 7 is accessed over address line 17 and its contents read by target CPU 1 over path 19. However, during realtime instruction translation, data cache 7 writably terminates instruction streams from source CPUs. This means that the data cache is the node from which the source instruction streams are accessed. In this regard, an IBM System 370 CPU is an illustrative multifield instruction stream source whose instructions can be locally stored in data cache 7. A complete description of IBM 370 host architecture is set out in G.M. Amdahl et al, U.S. Patent 3,400,371, issued 3 September 1968. The 3,400,371 patent is incorporated by reference.

An apparatus embodiment of the invention is in the form of an emulator assist processor (EAP) 3 accessing data cache 7 by way of address path 17a and read path 19a and the instruction cache 5 by way of address path 9 and read path 11a. The conversion output from the EAP is to target CPU machine 1 over path 15, merge 13, and line 21.

With these factors in mind reference should be made to FIG. 1 depicting the fields of an IBM 370 CPU instruction and its general mapping relations to a target machine instruction. Instructions in the

IBM 370 System computers consist of 2, 4, or 6 bytes and can contain up to 3 addresses. Five distinctive formats are used depending on the location of various operands required. The formats include:

1. RR (register/register) instructions. The operands R_1 and R_2 are CPU general registers. The result is placed in R_1.

2. RX (register/index) instructions. A first operand is located in R_1 while the other is in main memory. The effective memory address is $X_2 + B_2 + D_2$ where X_2 and B_2 denote the contents of general registers being used as index and base registers respectively, and D_2 is a relative address or "displacement" contained in the instruction. The result is placed in R_1.

3. RS (register/storage) instructions. Two operands are in general registers, a third is in main memory.

4. SI (storage/immediate) instructions. In this case, one operand is in main memory while the other is located within a predetermined range of contiguous bit positions of the instruction itself. This is an immediate operand as opposed to the usual operand address.

5. SS (storage/storage) instructions. Both operands are in main memory. The address as specified by the instructions are typically the initial addresses of two operand fields whose length is L bytes.

With reference to FIG. 1, the 370 instruction depicts an operation code field, typically of one byte followed by a pair of operands L1, L2 and a pair of base-plus-displacement addresses, namely B_1, D_1 and B_2, and D_2. These are to be mapped into a target machine instruction of 32 bits. The target instruction format includes an OP code field occupying bit positions 0—5, an RT field designating the register used to receive the result of an instruction in the positions 6—10, while the RA field in positions 11—15 is the name of the register used for the first operand. Depending on instruction type, the second half of the isntruction could include, in positions 16—20, the name of the register used as a second operand, in positions 21—25 the immediate field specifying the operation to be executed by a controller named in an adjacent field of bit positions 26—29. The remaining bit position contents define internal bus operation instructions.

Referring now to FIG. 3, when taken together with FIG. 2, it is apparent that when data cache 7 is addressed over path 17a, the contents consisting of a source instruction, are transmitted over path 19a and loaded into register 25. The OP code of the source instruction, accesses instruction cache 6 by way of address register 27 actuating path 9a. Responsively a microinstruction control section is transmitted to register 23 over path 11a. Each microinstruction may cause a subsequent microinstruction to be accessed so that each source instruction is replaced by an EAP microcode routine. A microcode instruction consists of a control section and a skeleton target instruction. The skeleton target instruction may have zeroed and/or meaningful register and displacement fields. Control information specifies how fields from the source instruction should be merged into the zeroed fields of the skeleton instruction by the EAP. During emulation, the EAP passes these completed target instructions to the target CPU to be executed. The target CPU executes these instructions normally, except that its instruction address register (not shown) remains fixed and the target CPU makes no attempt to fetch instructions. This parenthetically is termed cycle stealing. During emulation, the target CPU waits for the EAP to give it instructions to execute instead of fetching instructions itself. One way of terminating the translation for any specific source instruction can be upon EAP detection of a zeroed instruction field or a stop bit embedded in a predetermined bit position within a microcode sequence.

In executing translation, the target CPU initializes the EAP registers 27. A suitable state change is made in the target CPU. The first source instruction is fetched into the EAP internal register 25. The OP code portion of the source instruction forms the address to the first microcode instruction for this particular source instruction operation. The microcode instruction is then fetched from the instruction cache. The skeleton target isntruction portion of the microinstruction has its zeroed fields filled in from the appropriate fields of 370 instructions. The completed target instruction is then sent to the target machine for execution. Each microinstruction may either link to another microinstruction to be so processed, or it may be the last of a series for the current source instruction. This process is singularly repeated for each 370 or source instruction that is fetched. Significantly, each valid target instruction requires a microcode instruction of two words from the instruction cache. These are the control word and the skeleton target instruction. These are fetched consecutively with the OP code selected contol word being first.

Referring now to FIG. 4, there is shown the emulator micro control section format. The format of the 32 bits that make up the control section is allocated as follows: OP is the command to be executed in the EAP, R is the substitution control for the RT and RA target machine register fields, D is the substitution control for the displacement field and the RB target machine register field, C controls the condition codes while NI is the address of the next instruction to be executed by the EAP. If NI is 0, then the EAP will fetch and emulate the next System 370 instruction from the data cache, otherwise it will access the instruction cache again according to the content of the NI. This is aptly drawn in the FIG. 6 enhancement of the EAP 3 shown in FIG. 3. Note in the micro instruction formatted at register 23 in FIG. 6, an alternative to a 0 next instruction address for terminating the EAP fetch from the instruction cache 27 can be by way of a LAST bit position which is set when the last instruction has been fetched in a sequence from the instruction cache.

Referring now to FIG. 5, there is shown a timing diagram of the major reformatting operations of the EAP in overlap relation (pipelining) to increase throughput. While such pipelining is not the object of this invention, it is evident that significant performance throughput can be obtained.

The register transformation technique between the source and target register spaces provides significant performance gains. For example, because of the pipelining and merging of reformatted instructions from the EAP into the target machine instruction stream, the target machine which might normally execute instructions only every other cycle would permit execution to take place every cycle. This permits the EAP to cause repetitive functions to be executed in the target machine at the full execution rate.

Advantageously, the EAP can be operated in a subroutine mode whenever a sequence of source instructions do not require register space mapping. In this mode, the EAP receives regular target machine instructions, instead of micro-instructions from the instruction cache and the target machine again runs at full speed. The subroutine mode is terminable when the target machine is asked to execute an instruction which indicates the resumption of translation mode.

The embodiment heretofore described presupposes formation by the EAP of a complete target machine instruction by merging data extracted from the source instruction with the skeleton instruction from the instruction cache. This is illustrated in FIGS. 2, 3 and 6. One modification involves intercepting the skeleton target instruction and substituting fields in the complete target machine instruction before passing it into the target CPU, rather than merging it on the fly.

While the invention is particularly described with reference to a preferred embodiment it is to be appreciated that the method focuses on dynamic register field substitution on the fly. Source instruction strings generated from CPU's other than the IBM System 370 are certainly contemplated.

In order to avoid EAP bottlenecking, a multiple cache target machine is desirable for performance advantages.

Claims

1. Apparatus for translating multified source instructions into multifield target instructions in a target processor capable of executing target instructions and not source instructions by means of a self-controlling instruction translator logically situated between the instruction source and the target processor instruction execution element, characterised in that

a) the translator is micro-instruction controlled;

b) the translator is arranged to fetch a skeleton target instruction and one, or, sequentially, more than one, micro-instruction from memory from a location indexed by a pre-determined field of the source instruction currently being translated;

c) the translator is arranged, under control of the micro-instruction(s), to fill-in the skeleton target instruction by copying selected fields from the source instruction or by generating fields by computation from selected fields of the source instruction; and

d) the translator is arranged to insert the skeleton target instruction, once filled-in, into the target processor instruction stream.

2. Appratus as claimed in claim 1 wherein each fetched microinstruction includes an address portion which, if non-zero, designates the memory location of the next micro-instruction of a sequence of related micro-instructions or, if zero, indicates termination of the sequence.

3. Apparatus as claimed in either preceding claim wherein the translator is so arranged that the aforesaid operations of fetching, filling-in and inserting can be performed in an over-lapped manner.

4. Apparatus as claimed in any preceding claim wherein the target processor includes two caches in its memory system, the first of the caches containing data, from the point of view of the target processor, and the second containing instructions, the translator being arranged to fetch source instructions from the first cache and micro-instructions and skeleton instructions from the second cache, using cycle steal access to the second cache and to the instruction execution cycles of the execution element of the target processor.

Patentansprüche

1. Vorrichtung zum Übersetzen von Mehrfeld-Quelleninstruktionen in Mehrfeld-Zielinstruktionen in einem Zielprozessor, der fähig ist, Zielinstruktionen und nicht Quelleninstruktionen mittels eines selbstkontrollierten Instruktions-Übersetzers auszuführen, der logisch zwischen der Instruktionsquelle und dem Instruktionen ausführenden Element des Zielprozessors angeordnet ist, dadurch gekennzeichnet,

a) daß der Übersetzer durch Mikroinstruktionen gesteuert ist,

b) daß der Übersetzer ausgebildet ist, um eine Skelett-Zielinstruktion und eine oder sequentiell mehr als eine Mikroinstruktion aus einem Speicher aus einem Platz herauszuholen, der durch ein vorherbestimmtes Feld der Quelleninstruktion indexiert ist, die gegenwärtig übersetzt wird,

c) daß der Übersetzer ausgebildet ist, unter Kontrolle der Mikroinstruktion(en) die Skelett-Zielinstruktionen durch Kopieren ausgewählter Felder aus der Quelleninstruktion oder durch rechnerisches Erzeugen von Feldern aus ausgewählten Feldern der Quelleninstruktion zu ergänzen, und

d) daß der Übersetzer ausgebildet ist, die ergänzte Skelett-Zielinstruktion in den Zielprozessor-Instruktionsfluß einzufügen.

2. Vorrichtung nach Anspruch 1, dadurch gekennzeichnet, daß jede geholte Mikro-

instruktion einen Adressenteil aufweist, der bei Verschiedenheit von Null den Speicherort der nächsten Mikroinstruktion einer Folge von in Beziehung zueinander stehender Mikroinstruktionen bezeichnet, oder der, falls er Null ist, das Ende der Folge anzeigt.

3. Vorrichtung nach einem der vorstehenden Ansprüche, bei der der Übersetzer so ausgebildet ist, daß die vorerwähnten Operationen des Holens, des Ergänzens und des Einfügens in einer überlappenden Weise ausgeführt werden können.

4. Vorrichtung nach einem der vorstehenden Ansprüche, bei der der Zielprozessor zwei Cache-Speicher in seinem Speichersystem aufweist, wobei der erste der Cache-Speicher aus der Sicht des Zielprozessors Daten und der Zweite Instruktionen enthält, und wobei der Übersetzer so ausgebildet ist, um Quelleninstruktionen von dem ersten Cache-Speicher und Mikroinstruktionen sowie Skelett-Instruktionen vom zweiten Speicher zu holen, indem nach dem Cycle-Stealing-Verfahren ein Zugriff auf den zweiten Cache-Speicher und auf die Instruktions-ausführungszyklen des instruktionsausführenden Elementes des Zielprozessors erfolgt.

Revendications

1. Appareil pour la traduction d'instructions de source à champs multiples en instructions de cible à champs multiples dans un processeur de cible capable d'exécuter des instructions de cible et non des instructions de source, au moyen d'un traducteur d'instruction autonome situé logiquement entre la source d'instruction et l'élément d'exécution d'instruction du processeur de cible, caractérisé en ce que:

a) le traducteur est commandé par microinstruction;

b) le traducteur est prévu pour chercher une instruction de cible squelette et une, ou, séquentiellement, plus d'une, microinstruction dans une mémoire à un emplacement indexé par un champ prédéterminé de l'instruction de source en cours de traduction;

c) le traducteur est prévu, sous la commande de la microinstruction ou des microinstructions, pour remplir l'instruction de cible squelette, par copie de champs sélectionnés de l'instruction de source ou par génération de champs par calcul à partir de champs sélectionnés de l'instruction de source; et

d) le traducteur est prévu pour insérer l'instruction de cible squelette, une fois remplie, dans le train d'instruction du processeur de cible.

2. Appareil suivant la revendication 1, dans lequel chaque microinstruction cherchée comprend une partie d'adresse qui, si elle n'est pas à zéro, désigne l'emplacement de mémoire de la microinstruction suivante d'une séquence de microinstructions associées ou, si elle est à zéro, indique la fin de la séquence.

3. Appareil suivant l'une quelconque des revendications précédentes, dans lequel le traducteur est prévu de sorte que les opérations précitées de recherce, remplissage et insertion peuvent être exécutées en chevauchement.

4. Appareil suivant l'une quelconque des revendications précédentes, dans lequel le processeur de cible comprend deux caches dans son système de mémoire, la première des caches contenant des données, du point de vue du processeur de cible, et la deuxième contenant des instructions, le traducteur étant prévu pour chercher des instructions de source dans la première cache et des microinstructions et des instructions squelettes dans la deuxième cache, par accès à vol de cycle à la deuxième cache et aux cycles d'exécution d'instruction de l'élément d'exécution du processeur de cible.

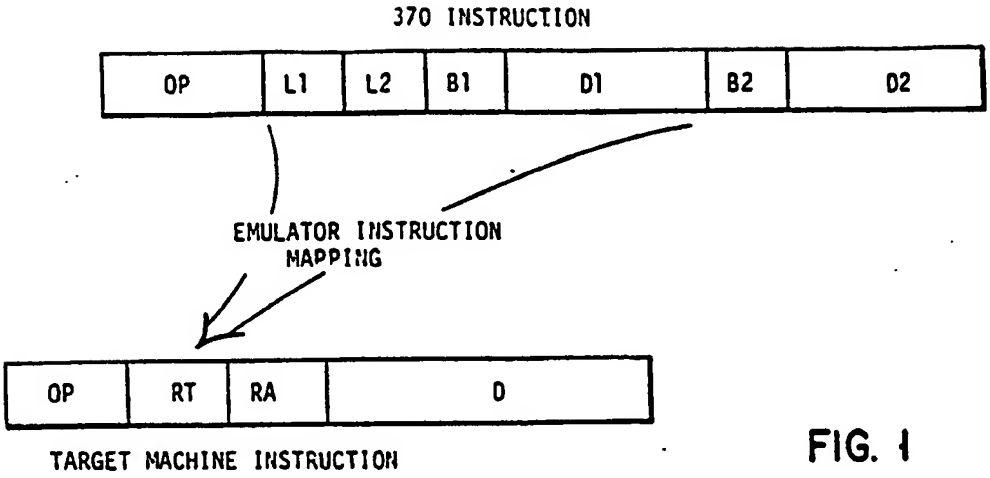

370 INSTRUCTION
OP
L1
L2
B1
D1
B2
D2
EMULATOR INSTRUCTION MAPPING
OP
RT
RA
D
TARGET MACHINE INSTRUCTION

FIG. 1

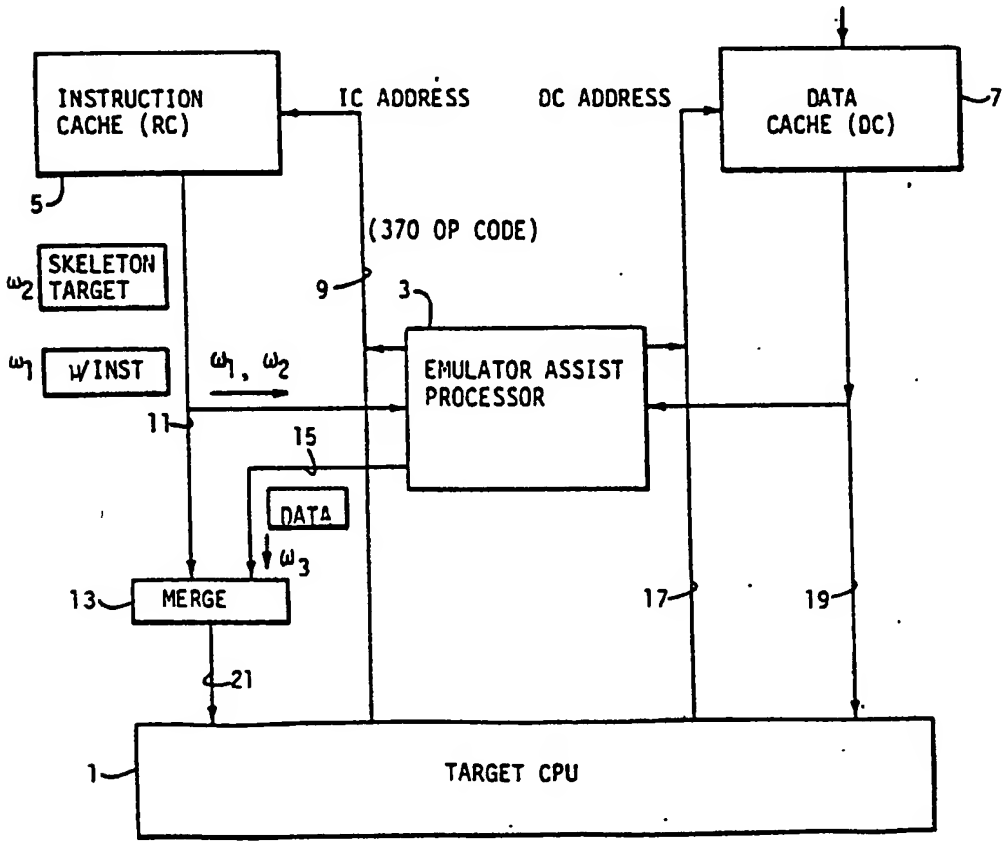

INSTRUCTION CACHE (RC)
IC ADDRESS
DC ADDRESS
DATA CACHE (DC)
7
5
(370 OP CODE)
SKELETON TARGET
ω2
9
3
ω1
W/INST
ω1, ω2
EMULATOR ASSIST PROCESSOR
11
15
DATA
ω3
13
MERGE
17
19
21
1
TARGET CPU

FIG. 2

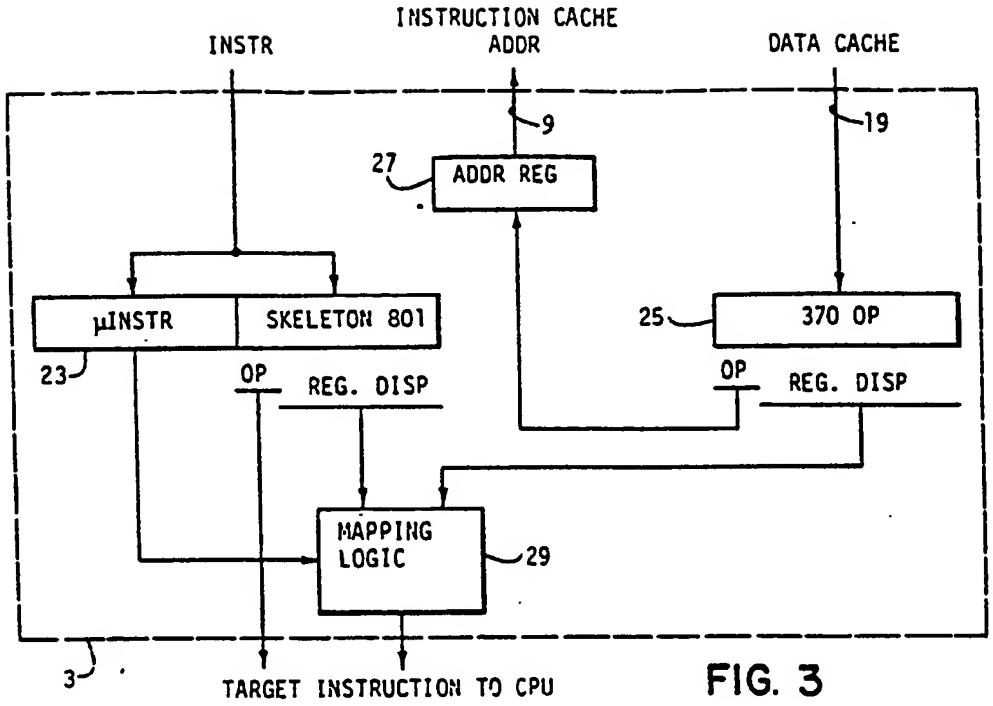
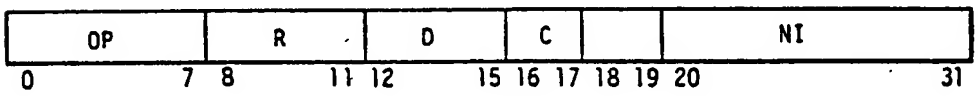


FIG. 3

370 EMULATOR MICROINSTRUCTION



OP	EMULATOR COMMAND
R	SUBSTITUTION CONTROL FOR TARGET MACHINE RT AND RA REGISTER FIELDS
D	SUBSTITUTION CONTROL FOR TARGET MACHINE RB AND DISPLACEMENT FIELDS
C	CONDITION CODE CONTROL
NI	NEXT INSTRUCTION ADDRESS

FIG. 4

OPERATION	MACHINE CYCLE							
	1	2	3	4	5	6	7	8
FETCH 370 OP	A							
FETCH μ INSTRUCTION			A		B		C	
FETCH 801 SKELETON				A		B		C
MAP 370 TO TARGET MACHINE					A		B	
EXECUTE TARGET MACHINE CODE WORD						A		B

FIG. 5

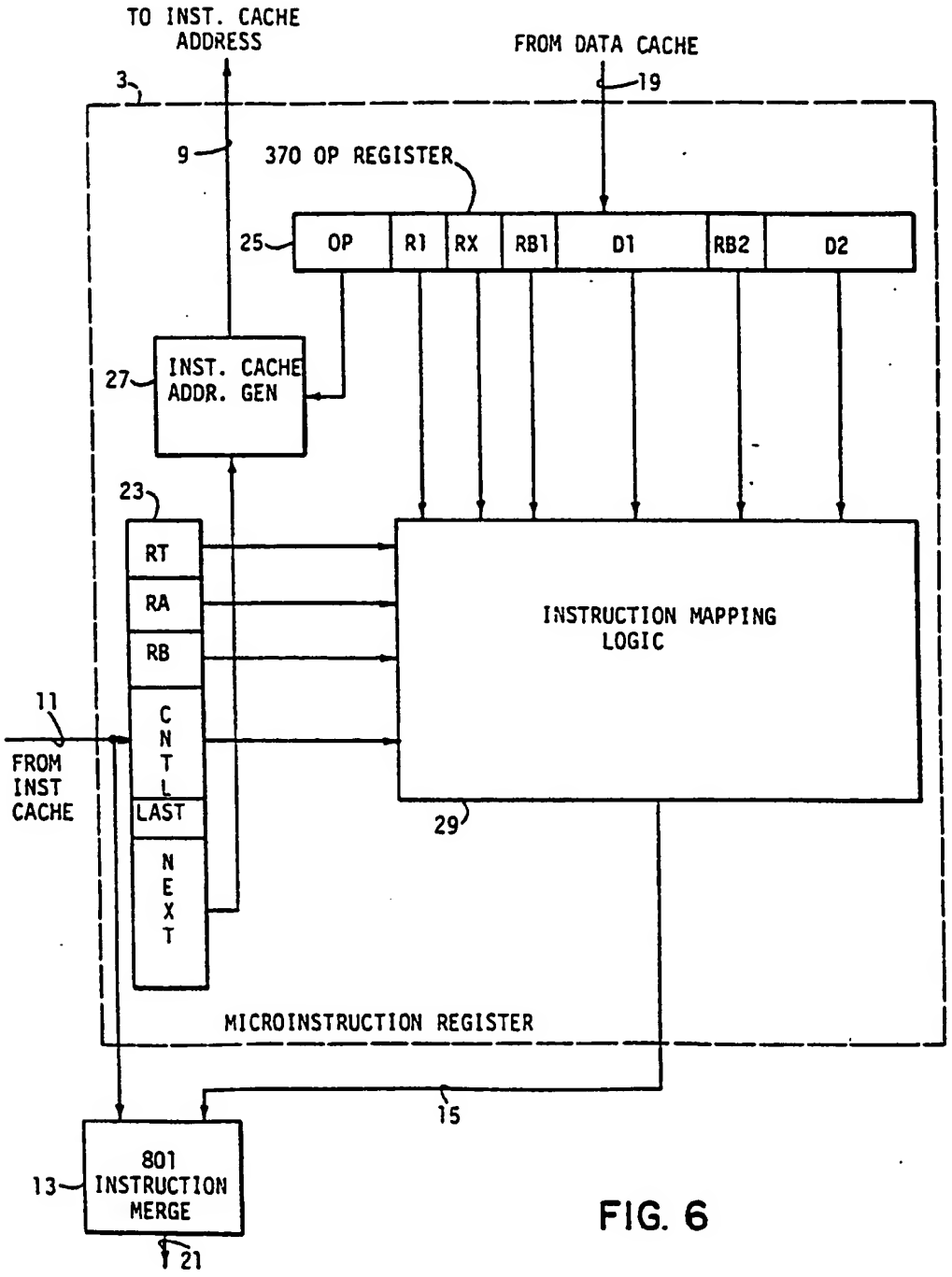

TO INST. CACHE ADDRESS
FROM DATA CACHE
3
19
9
370 OP REGISTER
25
OP
R1
RX
RB1
D1
RB2
D2
27
INST. CACHE ADDR. GEN
23
RT
RA
RB
CNTL
LAST
NEXT
INSTRUCTION MAPPING LOGIC
11
FROM INST CACHE
29
MICROINSTRUCTION REGISTER
15
13
801 INSTRUCTION MERGE
21

FIG. 6